Generating Repudiable, Memorizable and Privacy Preserving Security Questions using the Propp Theory of Narrative

Lei Pan
School of Information Technology
Deakin University
Melbourne, VIC, Australia
Email: l.pan@deakin.edu.au

Shaun Bangay
School of Information Technology
Deakin University
Melbourne, VIC, Australia
Email: shaun.bangay@deakin.edu.au

***Abstract*—Security questions are often based on personal information that is limited in variety, available in the public record and very difficult to change if compromised. A personalized folktale shared only by the communicating parties provides memorizable basis for individualized security questions that can be readily replaced in the event of a security breach. We utilize the Propp theory of narrative to provide a basis of abstraction for story generation systems. We develop a proof-of-concept system based on placeholder replacement to demonstrate the generation of repudiate and memorizable questions and answers suitable for online security questions. A 3-component protocol is presented that demonstrates the use of this process to derive a shared secret key through privacy amplification. This combination of story generation and communication security provides the basis for improvements in current security question practice.**

I. Introduction

Reeder and Schechter [18] identify several causes leading to failures of currently used security questions:

- The context of security questions does not apply to a user.
- Users are likely to forget the content or formatting of the answer that was selected at setup time.
- The adversary may have partial knowledge of user's personal information.
- The correct answers are highly guessable because they are common knowledge.
- The correct answers are researchable because they are found online or by asking someone who knows the answer.

Some of the above causes are mutually effected, and many researchers might overlook that the setup of security questions and the selection of answers are a form of chicken-egg problem. Most readers will have seen at least one of these questions — "*What is your mother's maiden name?*", "*What was the name of your third grade teacher?*", "*In what city did you meet your spouse?*", and "*What was the name of your first pet?*". More pessimistically, many sites reuse variations on a small set of similar questions. While some service providers allow their users to freely set their own security questions many users use personal information as answers to these questions. In principle, poorly designed security questions often suffer from high false positive and false negative rates — the former is caused by "low entropy or public availability of information" [8], and the latter is caused by ambiguity [20]. Hence, highly usable security questions and their answers must protect user's privacy and security according to Just [9].

A user's personal information can be collected or inferred from available public records. For instance, Maltego [16] and Social Engineering Toolkit (SET) [11] enable villains to collect personal information from *Facebook*, *Twitter*, *eBay*, *Amazon*, *YouTube*, blog sites, and other sites. Moreover, old tricks such as dumpster diving, wardriving and phishing still post persistent threats to private information. Furthermore, personal information cannot be revoked, repudiated or replaced if there is any compromise of the site containing stored answers.

With the increasing deployment of security questions, the protection of these security questions and users' answers are unclear. "During this lost decade, little innovation occurred in addressing cyberthreats while the adversary has significantly outpaced cybersecurity defenses to the point where most are obsolete", according to Gosh and McGraw [4].

In this paper, we propose an approach to generate and apply repudiable and privacy preserving security questions. In particular the answers should be memorizable given that security questions are typically invoked as a last resort. We assume that no extra security device or biometric information is used in this context. Adversaries possess similar context to that of the user and are capable of eavesdropping and injecting information on communication channels. Instead of introducing any new cryptographic scheme, we use the existing universal hashing protocol in the privacy amplification context [13] with necessary modifications. Therefore, our research question is composed of the following issues:

- How do we generate repeatable questions and answers without using any personal information?
- How do we assist ordinary Internet users to memorize a series of questions and answers?
- How do we secure the exchange of shared secret between a user and a server?

To address the above issues, we generate folktale-like stories which are made up of disposable and non-sensitive information. We procedurally generate security questions and answers based on the generated stories so that the user needs to only recall the content of the background story. We achieve transmission of secrets over public channels in the presence of adversary. Hence, our contributions are three-fold:

- A narrative generation approach that creates a distinctly unique story for each user, with memorizable elements and can be represented in a compact form.
- Mechanisms for question generation that can be customized to produce levels of information suitable for user validation and secret key exchange.
- A case study containing the process of integrating a background story, security questions and an extended modification of a well recognized secret key sharing protocol.

This paper is organized as follows: Section II surveys the existing literature to justify the research opportunity and identify relevant technologies; Section III introduces the Propp theory of narrative used for story representation and generation to the readers who are unfamiliar with story generation. Section IV proposes a three-component protocol — the first component generates a background story by using the Propp theory of narrative [17]; the second component derives a random list of security questions and expected answers based on the background story; and the third component uses the user's answers as basis of authentication and secret exchange by extending a universal hashing protocol introduced by Maurer and Wolf [13]. Section V presents a case study demonstrating the application of our protocol and significant conclusions are provided in Section VI.

II. Literature Review

Jakobsson et al. [8] advocate that "Good security questions should be based on long-lived personal preferences and knowledge, and avoid publicly available information." Though the questions from online matchmaking sites are effective to be used as security questions concluded in [8], the compromise of their answers would came at a great cost to users. To overcome this shortcoming and maintain the effectiveness of security questions, we propose to use generated stories containing no personal information as the source of security questions.

Story generation research drives understanding of the structure of narrative and provides procedural content generation for interactive fiction and video games. Several strategies that are used to generate stories include planning approaches [6], [19] that develop world and character state to meet defined goals, recombination of existing story elements [3], [6] and grammatical approaches [7] that reconstruct fresh narrative from an abstract representation of story structure. Grammatical approaches are well suited to rapid generation of stories. The morphology developed by Propp abstracts a common structure to Russian folktales and is frequently used by story generation systems [3], [7]. Case based reasoning is proposed in the *ProtoPropp* system [3] to select and adapt stories that are similar in a Propp function based feature space.

Explicit use of Propp's morphology is present in several online story generators including the Proppian Folktale Outline Generator [23], on which our system is based. This generator encodes the dependencies between functions but only returns the function listing without instantiating the elements of the story. A key technique of text generation is substitution. In particular, Topkara et al. [24] apply the synonym-substitution method to select and replace phrases of natural language text by predefined codewords.

Recall of specific items of information uses memorization strategies or mnemonics that encourage long-term retention. Stories and particularly folk-tales exhibit mnemonic resilience [15] by being passed down verbally across generations. Semantic encoding training such as sentence and story generation improves recall ability [12]. Story narratives are an effective way of improving recall of word lists even days after the original sighting [7]. In the context of information security, a positive correlation between memorability and education level has been reported [10].

Distinctive imagery increases memorability. Bizarre imagery, particularly in the context of common material, ensures that the concept remains distinct from other information in memory [14]. Use of these elements in counterintuitive ways should be avoided [15]. We apply this principle by using a familiar story structure with unusual and novel elements.

We need to protect the security of security questions and answers which are transmitted across the Internet. Perfect secrecy can only be achieved if the length of this key is at least as long as the message to be communicated on an untrustworthy message channel accessible to adversaries [21].

Privacy amplification is one of the recent breakthroughs in communication security. Formally, Bennett et al. [1] define privacy amplification as "the process that allows two parties to distill a secret key from a common random variable about which an eavesdropper has partial information". More specifically, Alice and Bob publicly agree on a function $h : \mathcal{S} \to \{0,1\}^r$ for a suitable r and compute the r-bit secret key $S' = h(\mathcal{S})$. Privacy amplification holds if the probability of selecting a function h with the following universal property —

Definition 1: A class $\mathcal{H}$ of functions $\mathcal{A} \to \mathcal{B}$ is universal if, for any distinct x_1 and x_2 in $\mathcal{A}$, the probability that $h(x_1) = h(x_2)$ is at most $1/|\mathcal{B}|$ when h is chosen at random from $\mathcal{H}$ according to the uniform distribution. [2]

In conclusion, to protect the users' secrecy and privacy whilst using security questions three components should be considered at once — a memorizable background story, security questions and answers derived from this story, and a communication protocol supporting privacy amplification.

III. Propp Theory of Narrative

Propp [17] identifies a structure common across an entire class of narrative; in this case the class of Russian folktale. Common plot pattern recur across many stories, each representing a common theme adapted to the context of the particular story. Key elements of Propp's abstraction are:

- An enumeration of the functions of the dramatis personae with respect to their roles in the story. The

actual instance of a particular role is the element that differs between stories. For example, a villain may be a dragon, group of bandits, witch or evil stepmother. Roles are also allocated to inanimate items that contribute to the story such as magical items that can assist in overcoming obstacles.

- A set of functions defining the acts of the characters that contribute significantly to the development of the story. For example, function 8: "*the villain causes harm to a member of the family*" applies when "*a dragon kidnaps the tsar's grandmother*". Instantiation of the function specifies the mechanism used to achieve that function.

Propp's analysis of Russian fairy tales shows that each progresses through a small sequence of states from a common set which still creates a wide variety of distinctive stories. This abstraction is limited to the folktale template. Similar abstractions exist for other genres (such as movie scripts [22]) and can be used to present stories in other multimedia formats [5].

Propp's theory [17] separates complex attributes of characters and functions such that character archetypes serve as stable elements in a tale, independent of how and by whom they are fulfilled. The number of functions is finite; specifically, Propp defines 32 story functions. Constraints have been developed based on analysis of the use of functions in existing folktales. For example, all stories provide the `initial situation` but only some will proceed through to the final `wedding` function (other may terminate when the hero dispatches the villain with extreme prejudice). Some functions operate in pairs: Function 23 "`unrecognized arrival`" of the hero in disguise is traditionally followed by an unmasking with function 27 "`recognition`". Some functions can be repeated; for example when the hero encounters and solves a sequence of difficult tasks. Generation of valid stories requires that these conventions be respected.

Through the above scheme, Propp suggest that "a wide range of substitution of certain variations for others can be ascertained on the whole" [17]. This philosophy of construction using substitution subject to constraints provides a foundation of our algorithm. The details of our story generator are in Section IV-A.

IV. Our Protocol

Our protocol consists of three parts — the first algorithm generates a background story B according to Propp theory; the second algorithm outputs a template of multiple choice questions denoted as T which consists of a list of questions Q and the predicted answers P; and the third algorithm is to distill shared secrets between user and server. Story is private to the client and server; security questions are private to the server, and a selection is revealed to the client on demand. The mapping of the three algorithms and corresponding sections is depicted in Figure 1.

The story generation algorithm is executed least frequently. A new story is created when a user establishes an online account on the server and shared between the user and the server at that time. A new story may be created if the server is

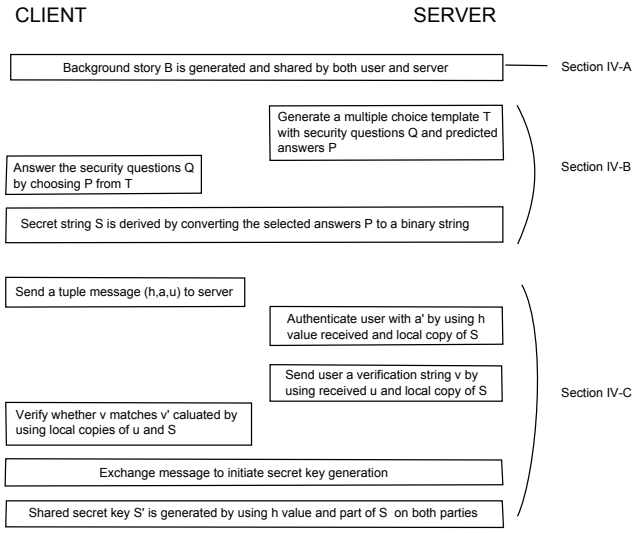

Fig. 1. The Framework of our Protocol

compromised and fresh security question context is required. The security question generation and secret sharing algorithms can be executed on demand. Moreover, the total secret string S is converted from the predicted answers P arranged in the queried order which is agreed by the user and the server. We denote the shared secret key as S'. In the cryptographic viewpoint, S is regarded as plain text and S' as encrypted cipher. According to information theory, the length of S' must be smaller than the length of S so that $len(S) - len(S')$ must be equal to the amount of information the adversary has about S plus a security parameter. The security parameter ensures the adversary's knowledge of S' less than the Rényi entropy of S. Consequently, the success probabilities of any impersonation attack or substitution attacks launched by the adversary are upper-bounded as shown in Definition 1.

A. Generating Background Story

A story is generated by the server and provided to the user securely. The server retains sufficient state information to recreate a set of questions and answers about the story that can be used as security questions. The user is expected to memorize the story so that the story content represents a shared secret. We use regular expressions for story generation using the structures identified under Propp theory. The generation process proceeds as follows: 1) A subset of the 32 Propp functions are selected for inclusion in the story. 2) A constraint checker ensures that dependencies between functions are satisfied. 3) A story element is chosen for each function. The story element consists of text describing the activity within the function. This text contains placeholders. 4) All placeholders are instantiated by choosing an element at random from an appropriate list. Once a placeholder is instantiated the same value is used for all subsequent substitutions. 5) The story is generated by concatenating the resulting story elements after placeholder substitution.

As an example, the functions chosen may include 0: `initial situation` and 2: `interdiction`. Associated story elements are: α_1: "*Once upon a time there was a $protagonist.*" and γ_1: "*The $family had always warned $protagonist:*

$interdiction.". These story elements contain 3 placeholders: `protagonist`, `family` and `interdiction`. If these are assigned values "brave, young girl", "elderly aunt" and "stay away from the sinister oak forest" then the resulting story will start with: "*Once upon a time there was a brave, young girl. The elderly aunt had always warned brave, young girl: stay away from the sinister oak forest.*"

This approach avoids the natural language generation problem encountered with grammatical and ontological based systems. We require a highly varied story both in terms of structure and of content. The narrative is confined to plot outlines typical of folktales. The regular expression mechanism produces a number of explicitly available values that can be queried in order to determine context shared between user and server.

The reader may notice the elaborate and quirky phrasing used in the values for placeholders in the examples shown. This is a deliberate attempt to enrich the information associated with each placeholder and the bizarreness; both intended to increase the memorability of the value.

We develop two principles to ensure that story elements produce relevant content when placed in the story: 1) Text for each story function should be stateless, in that almost any function may be the first line in the story. References to past events, other than through using placeholders should be avoided. 2) All placeholders relevant to that function should be present in the story text, since there is no guarantee that they occur elsewhere. This also ensures that all these elements are instantiated at this point and are available for use in questions about that function.

A single story represents shared state between user and server. This state can be completely defined by 1) the specifying the choice of functions used in the story, 2) which story elements are used in each function, and then 3) the value of the placeholders used across those textual elements. The text of the story can be reliably reproduced from this state information. We disallow repetition of story functions in order to avoid ambiguous stories or queries. This differ from the stories described by Propp theory [17] which enables repeated plot elements. We believe that duplication may incur user confusion of the story. That is, questions about that one instance of the function may overlap with other instances resulting in ambiguity with respect to the answer which contradicts our aim of generating memorizable security questions.

B. Generating Security Questions and Answers

Generated questions can query values related to particular dramatis personae or of the activity associated with an individual function. Thus a simple question queries the particular choice of placeholders in a specific story element. One of our generated security questions is "*What had elderly aunt always warned brave, young girl about?*" And the answer should be "*stay away from the sinister oak forest*" which is the value of the placeholder `interdiction`.

The number of functions used in the story can be chosen to ensure that there is sufficient content to generate a suitable number of security questions. We aim to generate a dozen or so security questions at a time, as indicated by experimental results in [8] where 10 to 16 security questions should be used to effectively distinguish legitimate users without incurring high false positive rate.

Our story generation algorithm in Section IV-A instantiates a series of Propp function f_i alongside with k_i placeholders. We use the following categories of security questions for each Propp function f_i:

1) The simplest type of query asks for the value of the placeholder at position j in function f_i. For example, given a story function f_i as "*The $protagonist proceeds to the lair of the $antagonist.*", the question can ask: "*Who went to the lair?*". In theory, we can construct maximally k_i unique queries in this category if the function description is sufficiently detailed.
2) More complex queries ask for the value of a placeholder in relation to the value for another placeholder. For example, given f_i as "*The $protagonist proceeds to the lair of the $antagonist.*", the question can ask: "*Whose lair did the $protagonist visit?*". In theory, we can construct maximally $k_i \times (k_i - 1)$ unique queries in this category.

We can obtain at least one security question from every Propp function. These questions are unique as long as each Propp function contains at least one placeholder. The number of questions available is within the range $[f, \sum_{i=1}^{f} k_i^2]$ for f Propp functions.

Each question answer is expected to be the value associated with a particular placeholder. When generating the questions, we instantiate placeholders containing the text for the question and answer. A series of incorrect answers are generated by querying the story generation system for other valid values of the answer placeholder. If n_i incorrect choices are created, then the answer to the question provides $\log_2(n_i + 1)$ bits of information. Furthermore, an unlimited number of security questions can be created from Propp functions using a diverse set of placeholders. In Section IV-C, we will use the answers to provide the secret string S as basis for authentication and as the input for privacy amplification.

C. Deriving Shared Secret Key by Using Privacy Amplification

By using the security question generation algorithm described in Section IV-B, the user and the server agree on a context string S which contains the information of the answers. Because of the potential leak of information by transmitting the string S over a public channel, we need to reveal least amount of information of the string S each time when it is used. We apply privacy amplification which is transforming the partially secret string into a virtually secret key S'.

Specifically, the server determines two values n and l such that 3 divides n, and l divides $2n/3$. Based on these parameters, the server generates a number of security questions whose answers can be represented as an n-bit string. After the security questions are chosen, the user will answer these questions so that the secret string S is agreed by the user and the server. Then, our shared secret key algorithm is defined as follows:

1) Both the user and the server split the n-bit string S into three parts S_I, S_{II}, S_{III} each of which has $n/3$ bits.
2) The user randomly proposes a privacy amplification seed $h \in GF(2^{n/3})$. And then the user calculates authentication string a by hashing this value and part of the secret string $a = h \cdot S_I + S_{II}$ in $GF(2^{n/3})$. Similarly, the user randomly generates a verification seed $u \in GF(2^l)$. After the three values h, a, u are ready, they are sent to the server.
3) The server receives the message sent in step 2. The server evaluates the authentication string $a' = h \cdot S_I + S_{II}$ by using received h value and the server's copy of S. The calculation is made in $GF(2^{n/3})$. If $a == a'$, then the server accepts the received h as the privacy amplification seed; otherwise, the user is notified to repeat step 2. Then the server calculates a verification string $v = f_u(S_I||S_{II})$ in $GF(2^l)$. The verification string v is sent to the user.
4) The user calculates $v' = f_u(S_I||S_{II})$ in $GF(2^l)$. If the received message v matches the calculated value v', then the user send a success message to the server. Upon the receipt of this message, the server sends an acknowledgement message.
5) Once the message in step 4 is acknowledged, both the user and the server derive the r-bit shared secret string by taking the r least significant bits of $h \cdot S_{III}$ in $GF(2^{n/3})$, that is, $S' = LSB_r(h \cdot S_{III})$; otherwise, the server and the user restart step 2.

Our shared secret key algorithm is similar to the Protocol UH (Universal Hashing) in [13]. Both algorithms transmit the same amount of information regarding h, a, u and v values. However, our algorithm differs from the Protocol UH in terms of the actions when validators mismatch and the time when the shared secret keys are derived. Our algorithm has advantages over the Protocol UH in the context of mutual authentication between the user and the server — 1) The lack of notification message defined in Protocol UH implies that communication party may enter into a non-deterministic state which may introduce severe security vulnerabilities. Our algorithm avoids this pitfall by introducing the error message which enables both user and server to fall back to a specified step if any error occurs. 2) We derive the shared secret key in the last step after the mutual authentication is accomplished between the user and the server; however, Protocol UH generates temporary shared secret keys before the mutual authentication, which requires extra effort in securely erasing the temporary keys.

Additionally, the background story should be regenerated given one of the following conditions —

- Either the server or the client requires to repudiate the shared secret which is currently used.
- The security questions have been exposed to users completely or over a predefined threshold.
- The server has been compromised.

This section defined our protocol. To demonstrate its application, we apply our protocol in a case study in Section V.

V. A Case Study

Supposing that a server determines $n = 24$ and $l = 4$, we distill a $r = 6$ bit shared secret key. Based on these parameters, the server generates a story containing the placeholder values listed in Table I.

Placeholder	Value
`king`	"tsar"
`possessivepronoun`	"her"
`pronoun`	"she"
`family`	"$elders"
`elders`	"grandmother and grandfather"
`venue`	"a magical garden"
`lossaction`	"perform community service"
`royalreward`	"a job in upper-management"
`protagonist`	"brave, young girl"
`conflict`	"a wet T-shirt competition"
`home`	"the royal palace"
`reward`	"lots of hugs and kisses"
`excursion`	"were collecting wood in the dark forest"
`antagonist`	"dragon"

TABLE I. Placeholder values of our generated story

Our story generating algorithm in Section IV-A outputs the following story according to the above placeholders and their values: *Once upon a time there was a tsar and the tsar lived with grandmother and grandfather in the royal palace. The grandmother and grandfather were very precious to the tsar. The grandmother and grandfather were collecting wood in the dark forest. The grandmother and grandfather decide to stay overnight in a magical garden. The grandmother and grandfather are kidnapped by a dragon. The call for a hero is issued. A brave, young girl responds to the call for help. The brave, young girl proceeds to the lair of the dragon. The brave, young girl engages in a wet T-shirt competition with the dragon. The dragon is defeated and had to perform community service. The grandmother and grandfather was freed and gave the brave, young girl lots of hugs and kisses. The brave, young girl accompanied the grandmother and grandfather on their return to the royal palace. The grateful tsar rewarded the brave, young girl with a job in upper-management.*

Following our question generation procedure, we obtain 12 randomly generated questions and their correct answers. The 12 questions, the correct answers and the associated placeholders are listed in Appendix A. To demonstrate the derivation of the secret S without affecting readability, we list the multiple choice format of the first four questions Q1, Q2, Q3 and Q4 the correct answers of which are used to construct the 8-bit binary string S_I.

Q1 *Who were very precious to the tsar?*
- A wife, a son and a daughter
- B horse, a dog and a sparrow
- C grandmother and grandfather
- D five lovely mistresses

Q2 *Who were collecting wood in the dark forest?*
- A wife, a son and a daughter
- B three daughters
- C grandmother and grandfather
- D five lovely mistresses

Q3 *What excursion did the grandmother and grandfather go on at the beginning of the story?*
- A were collecting wood in the dark forest

B travelled to visit distant family
C took a trip to Disneyland
D went for a walk in the grounds

Q4 *Where did grandmother and grandfather decide to stay when they were collecting wood in the dark forest?*
A a grubby motel
B a magical garden
C a gingerbread house
D a far pavilion

The correct answers are C, C, A, and B. We translate the answers by replacing choice A with "00", B with "01", C with "10" and D with "11". Hence, our S_I becomes 10100001. Similarly, the subsequent two secret strings become $S_{II} = 00111101$ and $S_{III} = 01001011$. Thus the entire secret string S becomes 101000010011110101001011.

Now we show a case of successfully distilling and exchanging 6-bit shared secret between the user and the server.

1) The user randomly proposes a privacy amplification seed $h = 00000010$ in $GF(2^8)$. Then the user calculates the authentication string $a = h \cdot S_I + S_{II} = 01100100$ in $GF(2^8)$. Similarly the user randomly generates a verification seed $u = 1010$ in $GF(2^4)$. The user send the three values h, a, u to the server.
2) The server evaluates the authentication string $a' = 01100100$ by using received $h = 00000010$ and the server's copy of S_I and S_{II}. The calculated value matches the value a sent by the user. Then the server derives the verification string $v = f_u(S_I||S_{II}) = 1010$ by multiplying u and the server's copy of S_I and S_{II} in $GF(2^4)$. The verification string u is sent to the user.
3) The user verifies $v' = f_u(S_I||S_{II}) = 1010$ by multiplying u and the user's copy of S_I and S_{II} in $GF(2^4)$. Since the two verification strings v and v' are identical, the user sends a success message to the server before generating the shared secret key.
4) Once the user receives an acknowledgement message from the server, both the user and the server derive the 6-bit shared secret key $S' = LSB_6(h \cdot S_{III}) = 010110$ by multiplying h and their own copies of S_{III} in $GF(2^8)$.

This case study demonstrates sharing a secret key between the user and the server. Our protocol is light-weight — our background story is made of 14 placeholders; 12 security questions are created in multiple choice format; and a 5-step process is applied to derive a shared secret key. For demonstration purposes, we only generate a short key sequence which should be prolonged for real-world implementation. That is, the shared secret can be algorithmically scaled up to an adequate length by adding more placeholders, more security questions, or more incorrect choices in each security question.

In practice, the process may be prolonged due to the possibility that the user may provide wrong answers by accident or by malicious attacks. A suitable threshold value can be chosen in case the genuine user makes a few mistakes. An upper-bound of incorrectly answered questions can be set to flag attacks. A new background story can be generated once all the security questions are used. Furthermore, a replay attack can be prevented by regenerating the story after the security questions are used. Moreover, the prevention of man-in-the-middle attacks is beyond this paper's scope because the machine-level end-to-end communication can be protected by existing solutions such as SSL or TLS-based VPN tunnels.

VI. Conclusion and Future Work

We have identified the need to separate personal privacy from security questions. We propose a practical and effective solution to generate effective and reliable security questions by using fictitious information. To make such information memorizable, we take the advantage of the Propp theory of narrative to generate background stories and develop a regular expression based story generation algorithm that can be readily queried for security questions. By considering user's privacy and security at once, our protocol meet the usability requirements recommended by [9]. We extend the universal hashing protocol in [13] by introducing error and success messages and by generating the shared key only after the authentication and verification. Our protocol enhances mutual authentication and secret sharing between the user and the server with repudiable and memorizable content by integrating story generation with communication security.

Future work includes a user study evaluating the memorability of stories used in this context. This evaluates the issues involved in recalling significant events in plain text stories as opposed to memorizing long passwords. Specifically, we plan to improve the composition of security questions to balance usability and security. In the long term we would like to implement our protocol as a cloud-based secondary authentication mechanism.

References

[1] C.H. Bennett, G. Brassard, C. Crépeau, and U. Maurer. Generalized privacy amplification. *IEEE Transaction on Information Theory*, 41:1915–1923, December 1995.

[2] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer System Science*, 18:143–154, 1979.

[3] P. Gervàs, B. Dìaz-Agudo, F. Peinado, and R. Hervàs. Story plot generation based on CBR. *Knowledge-Based Systems*, 18:235–242, 2005.

[4] A. Ghosh and Gary McGraw. Lost decade or golden era: Computer security since 9/11. *Security Privacy, IEEE*, 10(1):6–10, 2012.

[5] K. Glass and S. Bangay. A method for automatically creating 3d animated scenes from annotated fiction text. *IADIS International Journal on Computer Science And Information Systems*, 4:103–119, 2009.

[6] R. Hervàs, R.P. Costa, H. Costa, P. Gervàs, and F.C. Pereira. Enrichment of automatically generated texts using metaphor. In *6th Mexican International Conference on Artificial Intelligence (MICAI-07)*, pages 944–954, 2007.

[7] R. D. Hill, C. Allen, and P. McWhorter. Stories as a mnemonic aid for older learners. *Psychology and Aging*, 6:484–486, September 1991.

[8] M. Jakobsson, E. Stolterman, S. Wetzel, and L. Yang. Love and authentication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 197–200, 2008.

[9] M. Just. Designing and evaluating challenge-question systems. *Security Privacy, IEEE*, 2(5):32–39, Sept 2004.

[10] K. Kato and V. Klyuev. Strong passwords: Practical issues. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on*, volume 02, pages 608–613, 2013.

[11] D. Kennedy. Social engineering toolkit. http://www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_%28SET%29, 2014. [Software].

[12] B.A. Kirchhoff, B.A. Anderson, D.M. Barch, and L.L. Jacoby. Cognitive and neural effects of semantic encoding strategy training in older adults. *Cerebral Cortex*, 22:788–799, 2012.

[13] M. Maurer and S. Wolf. Secret-key agreement over unauthenticated public channels — part iii: Privacy amplification. *IEEE Transaction on Information Theory*, 49:839–851, April 2003.

[14] M. A. McDaniel and G. O. Einstein. Bizarre imagery: Mnemonic benefits and theoretical implications. *Advances in psychology*, 80:183–192, 1991.

[15] A. Norenzayan, S. Atran, J. Faulkner, and M. Schaller. Memory and mystery: The cultural selection of minimally counterintuitive narratives. *Cognitive Science*, 30:531–553, 2006.

[16] Paterva. Maltego. http://www.paterva.com/web6/products/maltego.php, 2014. [Software].

[17] V. Propp. *Morphology of the Folktale*. University of Texas Press, 2 edition, 1978.

[18] R.W. Reeder and S. Schechter. When the password doesn't work: Secondary authentication for websites. *Security Privacy, IEEE*, 9(2):43–49, 2011.

[19] M.O. Riedl and C. Leon. Generating story analogues. In *Proceedings of the 5th Artificial Intelligence and Interactive Digital Entertainment*, 2009.

[20] S. Schechter, A.J.B. Brush, and S. Egelman. It's no secret. measuring the security and reliability of authentication via 'secret' questions. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 375–390, 2009.

[21] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[22] B. Snyder. *Save the Cat! The Last Book on Screenwriting You'll Ever Need*. Michael Wiese Productions, 2005.

[23] Stone Dragon Press. Russian folktale outline generator. http://www.stonedragonpress.com/vladimir_propp/propp_generator_v1.htm, 2014. [Online; accessed 10-May-2014].

[24] U. Topkara, M. Topkara, and M.J. Atallah. The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th Workshop on Multimedia and Security*, pages 164–174, 2006.

APPENDIX A
TWELVE GENERATED QUESTIONS AND THEIR ANSWERS

The following 12 questions are automatically generated by our Python script; the answers are retrieved from the corresponding placeholders —

Q1 "*Who were very precious to the tsar?*" The correct answer is "*grandmother and grandfather*" which is equal to the value of the placeholder `$family`.

Q2 "*Who were collecting wood in the dark forest?*" The correct answer is "*grandmother and grandfather*" which is equal to the value of the placeholder `$family`.

Q3 "*What excursion did the grandmother and grandfather go on at the beginning of the story?*" The correct answer is "*were collecting wood in the dark forest*" which is equal to the value of the placeholder `$excursion`.

Q4 "*Where did grandmother and grandfather decide to stay when they were collecting wood in the dark forest?*" The correct answer is "*a magical garden*" which is equal to the value of the placeholder `$venue`.

Q5 "*Who stayed in a magical garden?*" The correct answer is "*grandmother and grandfather*" which is equal to the value of the placeholder `$family`.

Q6 "*Who kidnapped the grandmother and grandfather?*" The correct answer is "*dragon*" which is equal to the value of the placeholder `$antagonist`.

Q7 "*Who was kidnapped by the dragon?*" The correct answer is "*grandmother and grandfather*" which is equal to the value of the placeholder `$family`.

Q8 "*Who responded to the call for help?*" The correct answer is "*brave, young girl*" which is equal to the value of the placeholder `$protagonist`.

Q9 "*How did the brave, young girl battle the dragon?*" The correct answer is "*a wet T-shirt competition*" which is equal to the value of the placeholder `$conflict`.

Q10 "*What did the dragon do after being defeated?*" The correct answer is "*perform community service*" which is equal to the value of the placeholder `$lossaction`.

Q11 "*What did the grandmother and grandfather give the brave, young girl for saving them?*" The correct answer is "*lots of hugs and kisses*" which is equal to the value of the placeholder `$reward`.

Q12 "*Who gave the brave, young girl a job in upper-management?*" The correct answer is "*tsar*" which is equal to the value of the placeholder `$king`.